zühlke
empowering ideas

Effizientes Suchen mit Apache Solr

Java User Group 2023

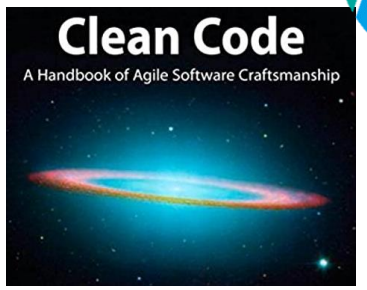
Matthias Graf | 12. Dezember 2023 | Bern

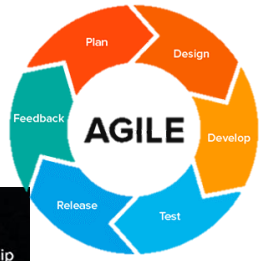
searching "hot dog"

google:

bing:

from https://www.reddit.com/r/dankmemes/comments/pi2aj3/why_is_my_search_engine_called_e621/





About Me

zühlke
empowering ideas

github.com/lizzyTheLizard/solr-jug2023

Folien, Code-Beispiele und Referenzen

Übersicht

Um was geht es heute?

Inhalt

- Was ist eine “Suche”?
- Was ist Solr?
- Wie kann ich Solr anpassen?

Out-Of-Scope

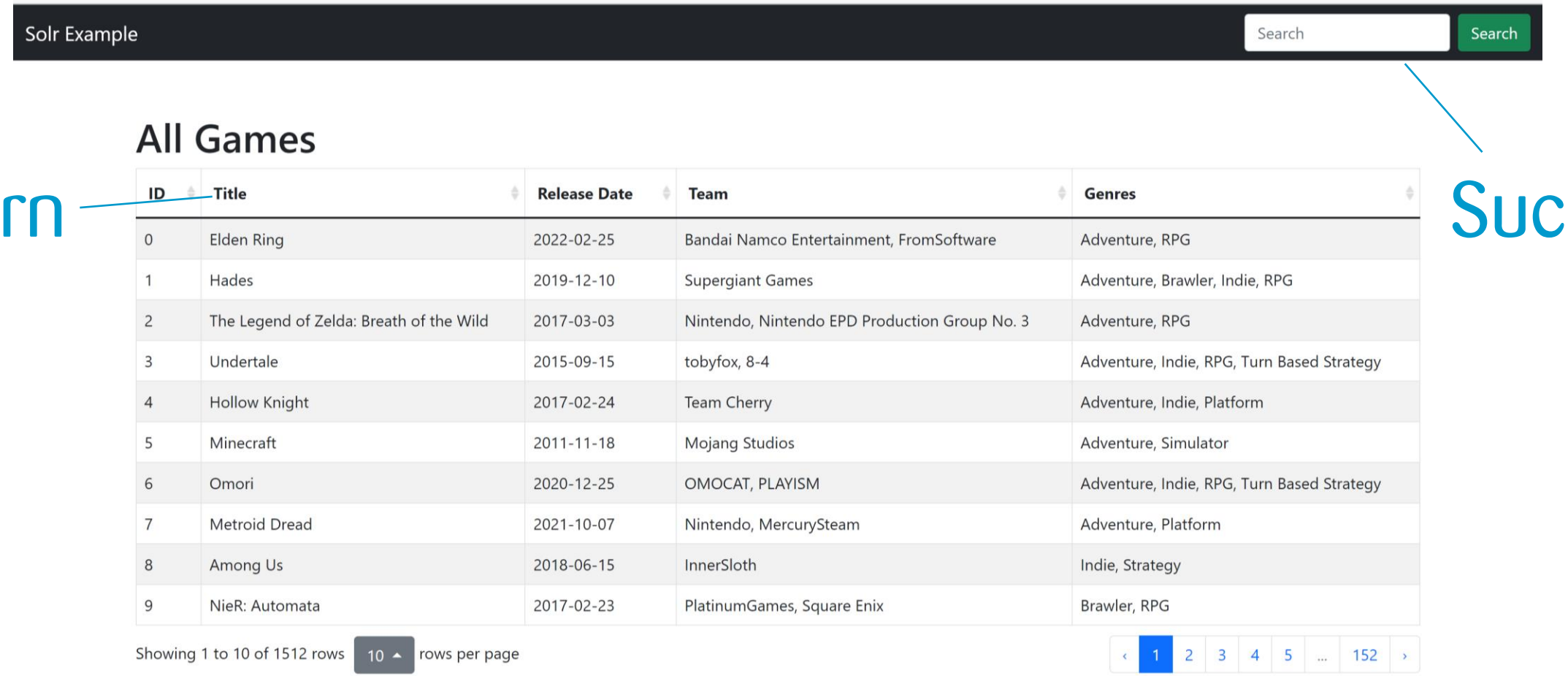
- Suche in Binärdateien
- Performance und Load Balancing
- Query-Languages

Ziel

- Wissen wann Solr sinnvoll eingesetzt werden kann
- Wissen wie Solr Integriert werden kann

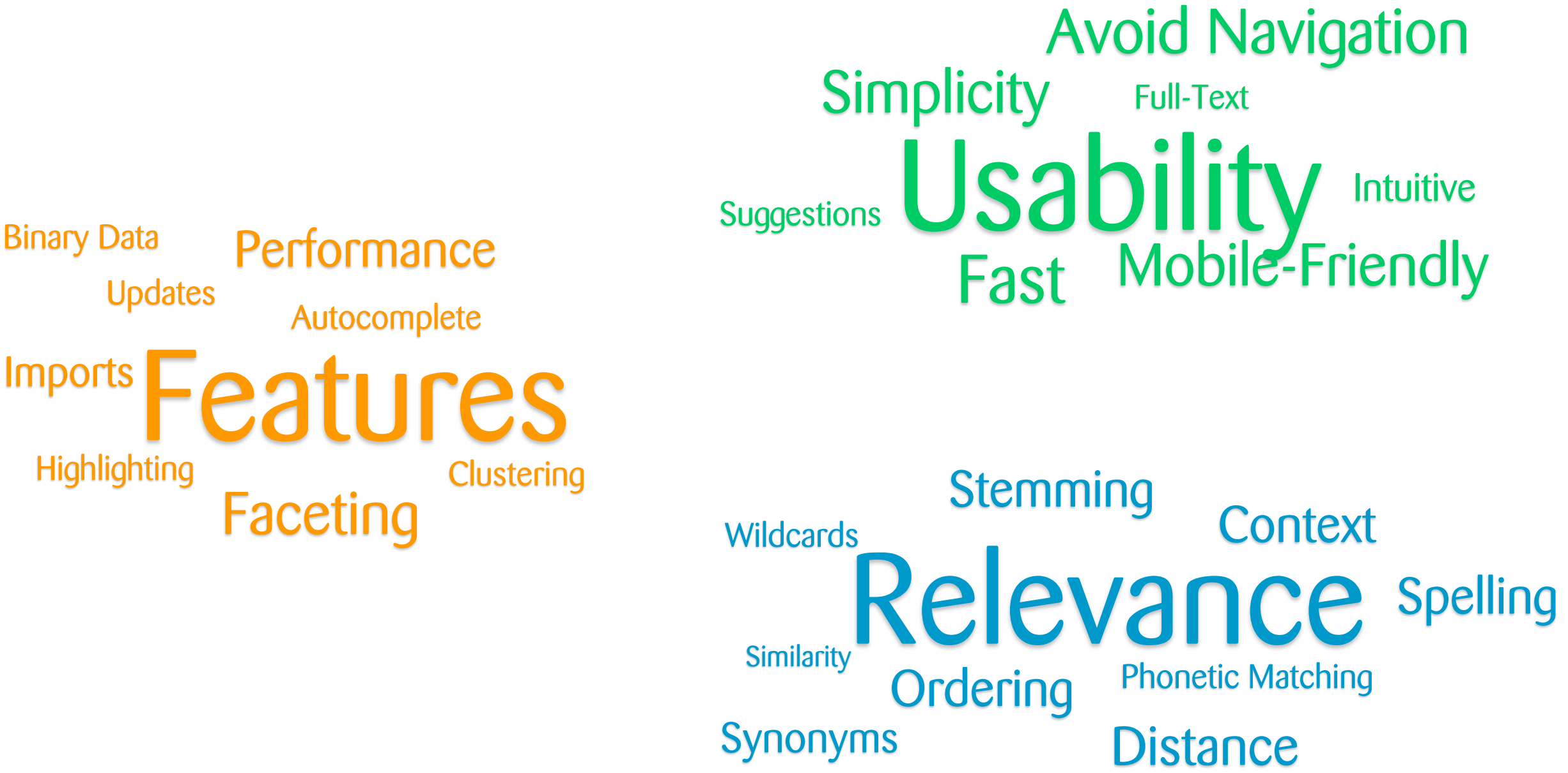
Was ist Suche?

Filtern oder Suchen?



Filtern

Suchen

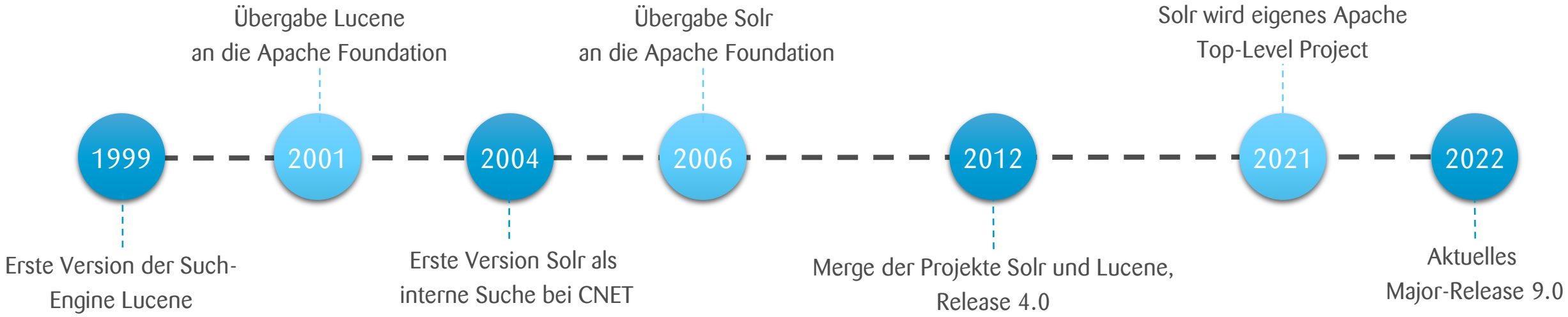

Avoid Navigation
Simplicity
Full-Text
Usability
Suggestions
Intuitive
Fast
Mobile-Friendly
Binary Data
Performance
Updates
Autocomplete
Imports
Features
Highlighting
Clustering
Faceting
Stemming
Wildcards
Context
Relevance
Spelling
Similarity
Ordering
Phonetic Matching
Synonyms
Distance

Was ist Solr?

Apache Solr

Facts and Figures

- Open-Source-Enterprise-Suchplattform
- Entwickelt in Java, basiert auf Apache Lucene
- Index-Suche
- Wichtigste Features: Volltextsuche, Highlighting, Faceting und Clustering, Replikation und Balancing, Binärdokumente etc.
- Weltweit in Verwendung unter anderem von DuckDuckGo, Adobe, Instagram, eBay, Netflix, Disney



Begriffe

Document	Eine Menge von Daten, die etwas beschreiben. Grundlegende Informationseinheit von Solr. Im Dokumente setzen sich aus Feldern zusammen (Semi-Strukturiert). Sollten über eine ID verfügen.
Field	Teil eines Document, können verschiedene Arten von Daten enthalten, z.B. Text, Fließkommazahlen, Listen etc.
Collection	Dokumente, die in einem einzigen logischen Index mit einer einzigen Konfiguration gruppiert sind.
Core	Eine einzelne Solr-Instanz / logischer Knoten. Mehrere Cores können auf einem einzigen physikalischen Knoten laufen.
Shard	Dokumente werden Shards zugewiesen. So kann eine Collection wenn nötigt auf mehrere Cores verteilt werden.

Was ist ein Index?

Spaghetti Carbonara
Id: 1
Zutaten: Spaghetti, Ei,…
Zubereitungszeit: 25 min
Zubereitung: Spaghetti im Salzwasser al dente garen. Zutaten mischen …

Erdbeertörtchen
Id: 2
Zutaten: Zucker, Ei, Sahne,…
Geräte: Backofen
Zubereitungszeit: 145 min
Zubereitung: Alle Zutaten für den Teig vorbereiten. Mehl, Salz und Zucker mischen …

Erbsen
Id: 3
Typ: Grundrezept
Zubereitung: Zwiebel in der heissen Bratbutter 5 Minuten anbraten. Mit Bouillon ablöschen und Erbsen beifügen…

Semistrukturierte Daten (Dokumente)

Indexing

Zubereitung
Zutaten.....1,2
Teig.....2
Vorbereiten.....2
Salz.....1,2
Mischen.....1,2
Spaghetti.....1
Al-Dente.....1
Garen.....1
Zwiebel.....3
Bratbutter.....3
Bouillon.....3
Erbsen.....3

Zubereitungszeit
145.....2
25.....1

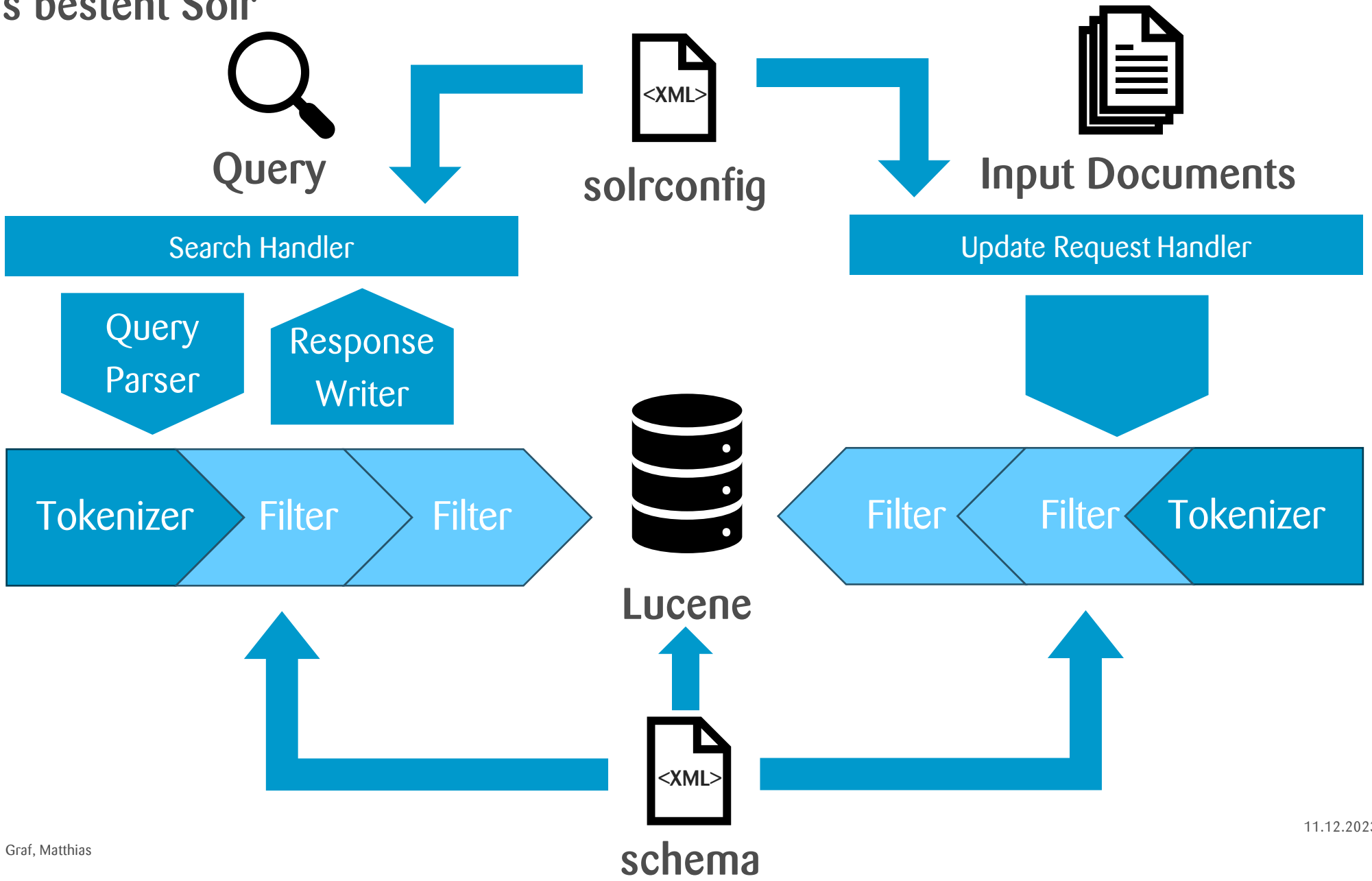
Typ
Grundrezept..3

Geräte
Backofen…..2

Zutaten
Ei.....1,2
Zucker……2
Sahne…….2
Spaghetti…1

Inverted Index

Aus was besteht Solr



Docker-Setup

Solr kann als Docker-Container gestartet werden

```
docker run -p 8983:8983 -t solr
```

Erweitertes Docker-Compose-Setup

```
solr:
    image: 'solr:9.2.0'
    ports:
      - "8983:8983"
    command:
      - solr-precreate
      - COLLECTION_NAME
    volumes:
      - ./src/main/solr:/opt/solr/server/solr/configsets/_default/conf
      - solr-data:/var/solr
```

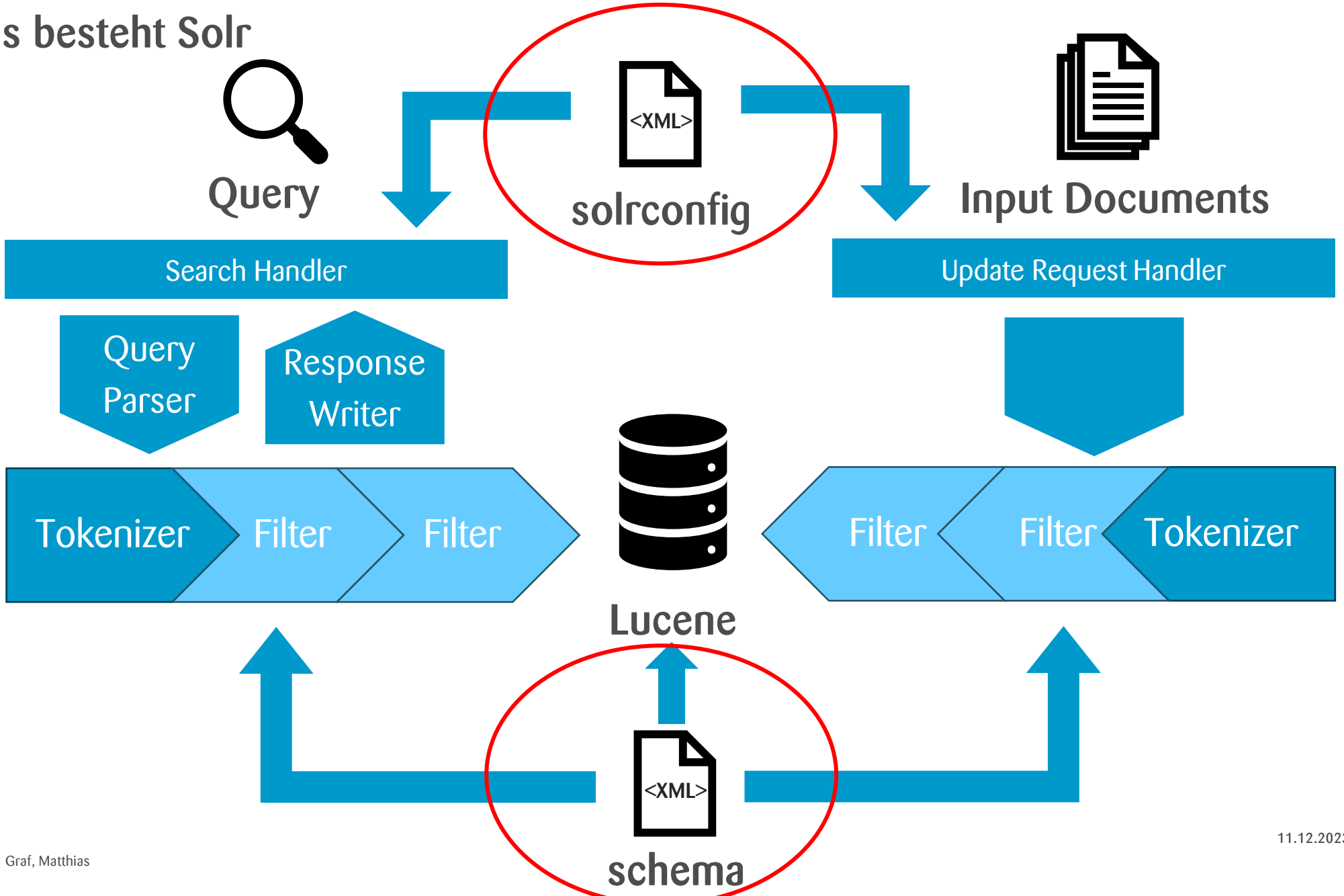
Live Demo

Zusammenfassung Live Demo

- Solr-Server starten (Container oder Binary)
- Eigene Collection erstellen (precreate-script oder über UI)
- ManagementUI localhost:8983/solr ausprobieren
- Integration in eigene Applikation mittels Client-Library (SolrJ) oder Rest-API
- Dokumente hinzufügen (Indexing)
- Suchen ausführen (Querying)

Solr Anpassen

Aus was besteht Solr



Default Configuration

- Solr stellt eine vollständige Default-Konfiguration zur Verfügung: Managed Schema (kann zur Laufzeit angepasst werden), DynamicFields mittels präfix, Field-Type Guessing
- Für Test und erste Erfahrungen reicht diese, für produktive Setups muss das Schema in der Regel aber angepasst werden

Default Konfiguration: github.com/apache/solr/tree/main/solr/server/solr/configsets/_default/conf

Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema name="example" version="1.6">
        <!-- Static filed definition-->
        <field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" />

        <!-- Dynamic field definitions allow using convention over configuration-->
        <dynamicField name="*_i" type="pint" indexed="true" stored="true"/>

        <!-- Field types define the type of a field e.g. the StrField type is not analyzed, but indexed/stored verbatim. -->
        <fieldType name="string" class="solr.StrField" sortMissingLast="true" docValues="true" />

        <!-- A text field with defaults appropriate for English-->
        <fieldType name="text_en" class="solr.TextField" positionIncrementGap="100">
                <analyzer type="index">
                        <tokenizer name="standard"/>
                        <filter name="stop" ignoreCase="true" words="lang/stopwords_en.txt"/>
                        <filter name="lowercase"/>
                        <filter name="englishPossessive"/>
                        <filter name="keywordMarker" protected="protwords.txt"/>
                        <filter name="porterStem"/>
                </analyzer>
                <analyzer type="query">
                        <tokenizer name="standard"/>
                        <filter name="synonymGraph" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
                        <filter name="stop" ignoreCase="true" words="lang/stopwords_en.txt" />
                        <filter name="lowercase"/>
                        <filter name="englishPossessive"/>
                        <filter name="keywordMarker" protected="protwords.txt"/>
                        <filter name="porterStem"/>
                </analyzer>
        </fieldType>
</schema>
```

Request Handler

```
<requestHandler name="/select" class="solr.SearchHandler">
        <lst name="defaults">
                <str name="echoParams">explicit</str>
                <int name="rows">10</int>
        </lst>
</requestHandler>


<initParams path="/update/**,/query,/select,/spell">
        <lst name="defaults">
                <str name="df">_text_</str>
                <str name="defType">edismax</str>
                <str name="qf">title^10 team^5 _text_^1</str>
        </lst>
</initParams>
```

Copy-Field

- Felder können mittels «Copy» mehrfach indexiert werden
- Dies ist z.B. sinnvoll wenn unterschiedliche Filter verwendet werden
- Haupteinsatz ist die Suche über verschiedenen Feldern

```
<field name="_text_" type="text_general" indexed="true" stored="false" multiValued="true"/>

<copyField source="title" dest="_text_" maxChars="30000" />
<copyField source="genre" dest="_text_" maxChars="30000" />
<copyField source="team" dest="_text_" maxChars="30000" />
```

Sorting and Scoring

- Jedes Resultat definiert einen «score» («je höher je relevanter»).
- Die Berechnung des Scores ist kompliziert und nicht immer einfach nachvollziehbar.
- Die Berechnung des Scores kann mittels verschiedener Parameter beeinflusst werden.
- Resultate können sortiert werden (sort-Parameter), nach Score, Feldern(*) oder Funktionen sortiert werden.
- Standartmässig ist Sortierung nach dem ‘score’.

Takeaways

Takeaways

- Eine Suchfunktion ist **wichtig**, eine gute Suchfunktion zu implementieren ist aber **schwer**
- Requirements für Suche in der Regel unklar -> **Start Slow and Improve**
- Eine wirklich gute Suche **braucht kontinuierlichen Aufwand**
  - Nach was sucht der Benutzer? Findet er die richtigen Resultate?
  - Wie benutzt der Benutzer die Suchfunktion?
  - Was für Daten werden durchsucht?
- Apache Solr ist eine Open-Source Lösung mit guter Performance, vielen Features, einfach zu integrieren aber schwer zu meistern

Tipps und Tricks

- Eine Collection für unterschiedliche Daten
- Möglichst alle Daten aufnehmen, Schema-Änderungen und Reindex ist teuer
- Persönlich: Bessere Erfahrung mit „nicht-gemanagten“ Schema („configuration-as-code“)
- Eigene Felder Definieren, von der Default-Konfiguration kopieren wenn möglich (Field-Types etc.), anpassen wo notwendig
- solrconfig.xml nur anpassen wenn notwendig, möglichst minimal. Ausnahme: requestHandler
- ‚fq‘-Parameter nutzen, um Resultate zu filtern
- ‚Score‘ verstehen, debugging mittels ‚`why_score:[explain style=nl]`‘ als ‚fl‘-Parameter
- Boosting mittels ‚qf‘-Parameter

Pain Points

- Berechtigungsprüfungen aus den Suchresultaten?
- Wann Index updaten? Hohe Last oder veraltete Daten?
- Sortierung der Resultate? Insbesondere bei verschiedenen Typen

Fragen?

github.com/lizzyTheLizard/solr-jug2023

Folien, Code-Beispiele und Referenzen

Weitere Informationen

solr.apache.org/guide

Ofizielle Solr Dokumentation; Tutorials, Guides für Deployment und Konfiguration, Dokumentation der Query-Parser und vieles mehr

hub.docker.com/_/solr

Offizielles Docker-Image von Solr

baeldung.com/apache-solrj

SolrJ-Tutorial